UNIVERSITY OF WEST LONDON
The Career University

TC70029E – Digital Signal Processing for Audio Applications

Assessment 3: DSP Effect Processor

June 2022

Harriet Rena Cerys Drury

21474551

Abstract

This technical report highlights the research, prototyping and implementation of a DSP effect processor, a digital emulation of the Electro Harmonix Big Muff Pi guitar pedal. Initial research includes circuit evaluation, distortion, tone and gain control.

Shelving filters are discussed in tandem with the filters. The methodology details the analog modelling of high pass and low pass filters, with the utilisation of soft clipping (cubic distortion), half wave rectification and hard clipping.

A MATLAB script prototype is successfully implemented to include three potentiometer controls, affecting sustain, tone and volume respectively.

The final implementation is as a MATLAB application, allowing for a graphic user interface and user controls. This implementation is seen as a success to the desired requirements, however, future work to implement real time changes and closer recreations of the tone and distortion stages are discussed.

Table of Contents

1. Introduction

This technical report discusses the theoretical and developmental efforts for the creation of a DSP effect. The Electro Harmonix Big Muff Pi is chosen as an analog inspiration for this digital emulation. Specifications are chosen from the main stages of the original circuit board. This includes input, clipping, filtering and gain sections.

The structure of the report is to highlight the relevant background knowledge with guitar pedal emulation, an overview of the big muff pi, clipping, filtering and gain research. Focus is placed on the relevant mathematics for distortion and shelving filters. The report then discusses the implementation of researched theorems into the programming application MATLAB, with a critical reflection and results discussed.

Guitar Pedal Emulation

Guitar pedal emulation is a common area of research within digital physical modelling of analog audio effects. This approach induces the usage of digital signal processing techniques to largely resemble the analog counterpart (Holters, Dempwolf and Zolzer, 2011). For this project, focus is placed onto the emulation of analog circuitry of the Electro Harmonic Big Muff.

The Big Muff Pi

The Electro Harmonix Big Muff pedal is commonly used as a fuzz effect for a guitarist's signal. With decades of usage by the guitarist with musicians such as Dave Gilmour and Mogwai (Rae, 2012). It provides a distorted tone for the guitarist, with three main controls; sustain, tone and volume. Figure 1.1 below is the casing of the pedal.

Figure 1.1 The Big Muff Pi Pedal (Electro Harmonix, 2022)

This provides the user the ability to change the potentiometers controlling gain, filtering and distortion respectively. Figure 1.2 provides a breakdown of the internal circuitry of the pedal, with sections highlighted to break down the functionality. Sections are broken down by colour, for ease of interpretation. From this breakdown we observe that there are five distinguishable sections; input, first clipping, second clipping, tone and output. For the digital modelling, focus is placed on the clipping sections, tone and output gain control.

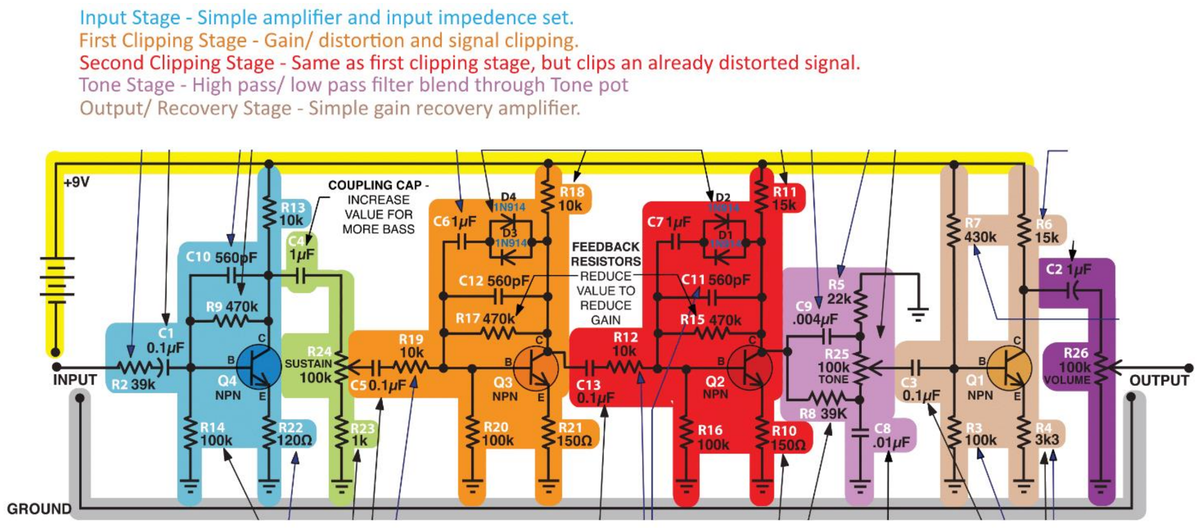

Input Stage - Simple amplifier and input impedence set.
First Clipping Stage - Gain/ distortion and signal clipping.
Second Clipping Stage - Same as first clipping stage, but clips an already distorted signal.
Tone Stage - High pass/ low pass filter blend through Tone pot
Output/ Recovery Stage - Simple gain recovery amplifier.
+9V
INPUT
OUTPUT
GROUND
COUPLING CAP - INCREASE VALUE FOR MORE BASS
FEEDBACK RESISTORS REDUCE VALUE TO REDUCE GAIN
R24 SUSTAIN 100k
R25 100k TONE
R26 100k VOLUME
Q1 NPN
Q2 NPN
Q3 NPN
Q4 NPN

Figure 1.2 The Circuit Board for the Big Muff Pi (Rae, 2012)

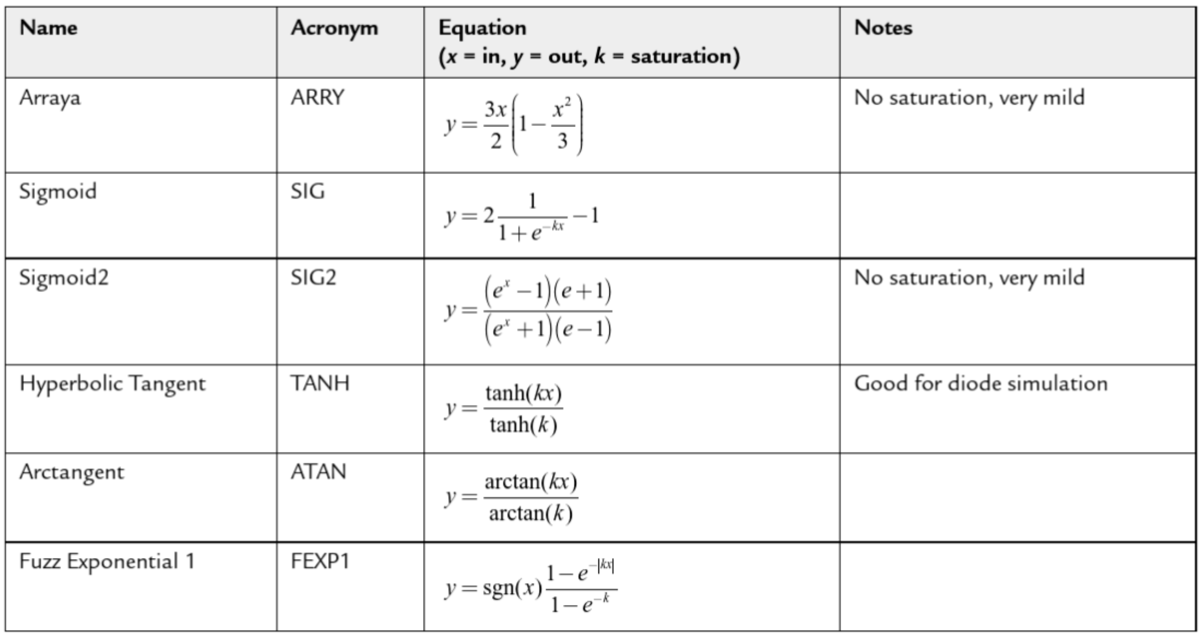
Clipping Stages

The clipping stage of this pedal invokes the distortion nature of the pedal. Clipping is a terminology used within the distortion effect branch, alongside overdrive. analog clipping in guitarist equipment pushed hardware components past their limit, resulting in the distorted sound (Russell, 2020) (Reiss and McPherson, 2015, pp.167–188).

The initial clipping stage utilises a soft clip, with a wave rectification clip used in a cascading sequence to affect the initially clipped signal.

Soft Clipping

Altering the amplitude of inputted waveform via a nonlinear smoothing curve alters the inputted signal, creating the auditory effect of distortion (Tarr, 2019, pp.159). Discrete time non linear processing can however cause aliasing issues, caused by the expansion of the signals bandwidth (Julius Orion Smith, 2011) (Reiss and McPherson, 2015, pp.167). Table 1 below highlights some wave shaping functions for soft clipping.



Name	Acronym	Equation **(*x* = in, *y* = out, *k* = saturation)**	Notes
Arraya	ARRY	$y = \frac{3x}{2}\left(1 - \frac{x^2}{3}\right)$	No saturation, very mild
Sigmoid	SIG	$y = 2\frac{1}{1+e^{-kx}} - 1$	
Sigmoid2	SIG2	$y = \frac{(e^x - 1)(e+1)}{(e^x + 1)(e-1)}$	No saturation, very mild
Hyperbolic Tangent	TANH	$y = \frac{\tanh(kx)}{\tanh(k)}$	Good for diode simulation
Arctangent	ATAN	$y = \frac{\arctan(kx)}{\arctan(k)}$	
Fuzz Exponential 1	FEXP1	$y = \operatorname{sgn}(x)\frac{1-e^{-\lvert kx \rvert}}{1-e^{-k}}$	

Table 1 Soft Clip Sigmoid Waveshaper Functions (Pirkle, 2019, pp.548)

These wave shaper functions are characterised by smooth approaches to the clipping level, creating a smoother, warmer sound in comparison to the hard clipping. The soft clip utilised in this application is the cubic distortion. Equation ?.? below is the function used on the input signal in the first clipping stage.

$$y[n] = x[n] - a \cdot \frac{1}{3} \cdot (x[n])^3$$

Equation 1. The Cubic Function (Tarr, 2019, pp. 160)

Where $x[n]$ is the inputted signal, $y[n]$ is the outputted signal and a is the amount of cubic distortion passing through the function. Zero would caused solely the x[n] signal be returned (no distortion), while one would allow for the whole cubic distortion to be factored into the signal (full distortion). This application utilises a static value of a at 1.

Figure 1.3 below is an example of the waveform and characteristic curve of the cubic distortion function. The characteristic curve highlights how amplitudes are curved instead of cut. The sine wave form sees a reduction in amplitude at lower frequencies.

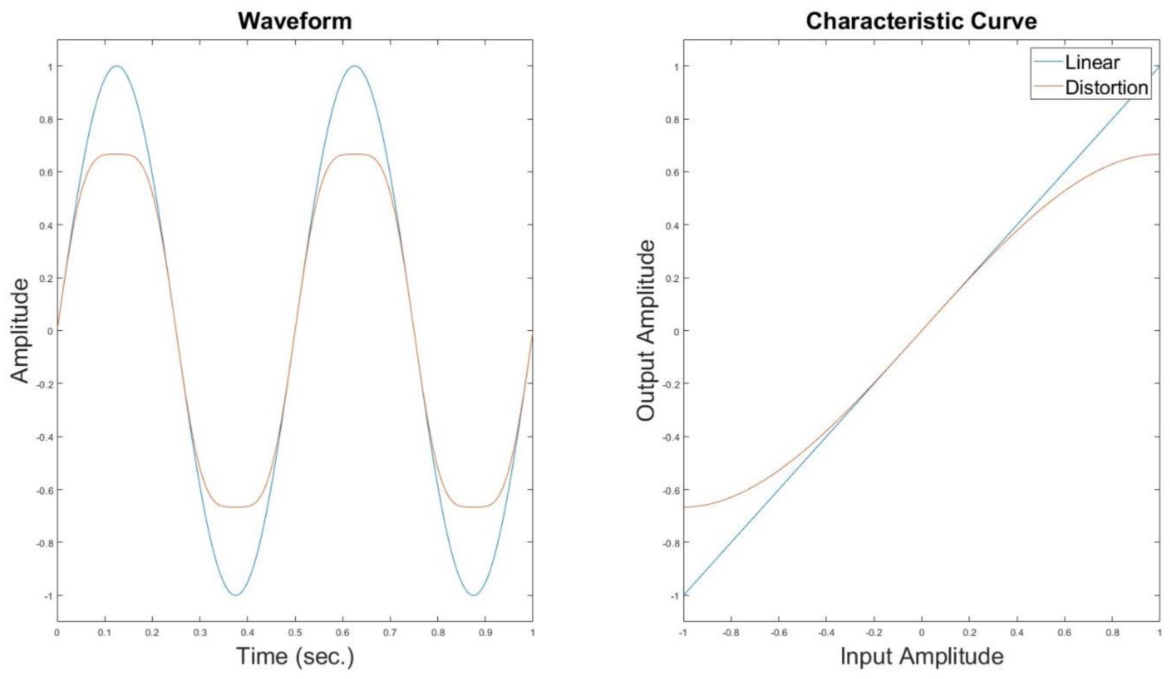


Figure 1.3 The Waveform and Characteristic Curve of Cubic Distortion

The Ibanez tube screamer guitar pedal utilises this type of effect within an analog version (Malaker, 2019). This creates the overdrive effect with a reduction in amplitude at lower frequencies noted.

Half Wave Rectification

For the second clipping stage, a hard clip was initially reviewed for the process. The hard clip is characterised by an abrupt transition between unclipped and clipped regions of the waveform (Reiss and McPherson, 2015, pp.169). This produces sharp corners on the waveform, producing a bright and harsh sound.

Rectification is the process of changing the negative values of a signal to be distorted in some way (Tarr, 2019, pp.189). Half wave rectification reduces all the negative amplitudes to zero. Equation 2 below is the threshold limits for half wave rectification.

$$y[n] = \begin{cases} x[n], & x[n] \geq 0 \\ 0, & x[n] < 0 \end{cases}$$

Equation 2. The Branch Equation for Half Wave Rectification Clipping

Where $y[n]$ is the outputted signal and $x[n]$ is the inputted signal. In contrast to full wave rectification, which changes the negative amplitudes to positive, half wave rectification creates a slightly less distorted output.

Figure 1.4 shows the waveform characteristics for this type of distortion. Full MATLAB code for the generated graphs are given in appendix 1.

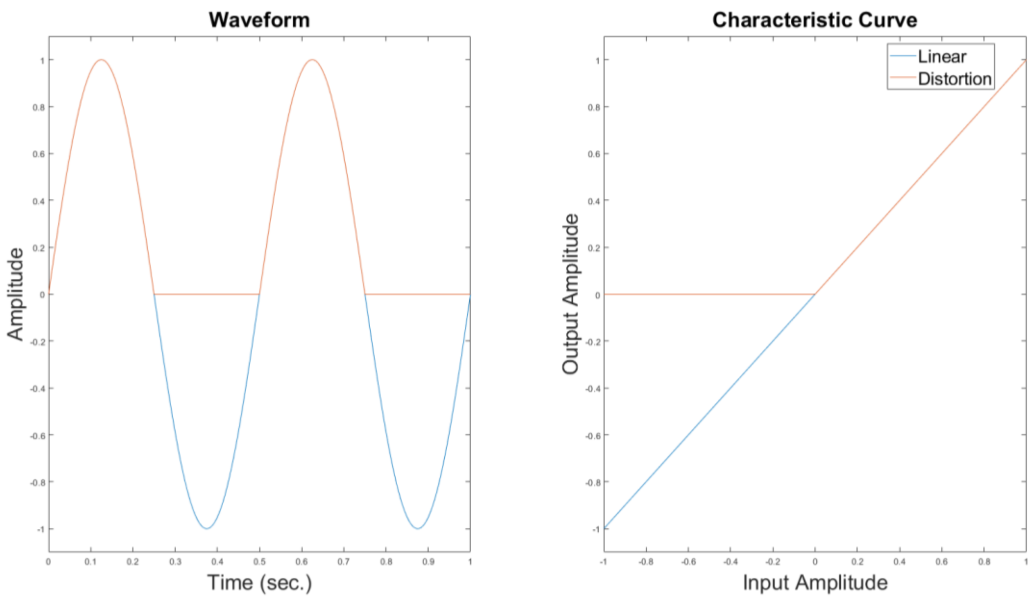


Figure 1.4 The Waveform and Characteristic Curve of Half Wave Rectification

For this application, both distortion types are applied in series. The graphic user interface allows for the user to control the amount of distorted signal vs original signal in the final mix. Changes to the amount of distortion are set manually within the program.

Hard Clip

The second portion of the second clipping stage is a hard clip threshold distortion (Tarr, 2019, pp.157). For this pedal emulation, the threshold is set for the user. If the amplitude of the signal goes over the specified signal, then it is cut to the threshold amplitude (Pirkle, 2019, pp.535). A branch equation for this is shown in equation 3 below.

$$c[n] = \begin{cases} thresh, & if\ x[n] > thresh \\ x[n], & if - thresh < x[n] < thresh \\ -thresh, & if\ x[n] < -thresh \end{cases}$$

Equation 3. The Branch Equation for Threshold Limiting Hard Clipping

Figure 1.5 highlights the characteristics of hard clipping onto a sine wave, hard clipping can be called distortion, and is famously used in analogue pedals such as the BOSS DS-2 (Malaker, 2019).

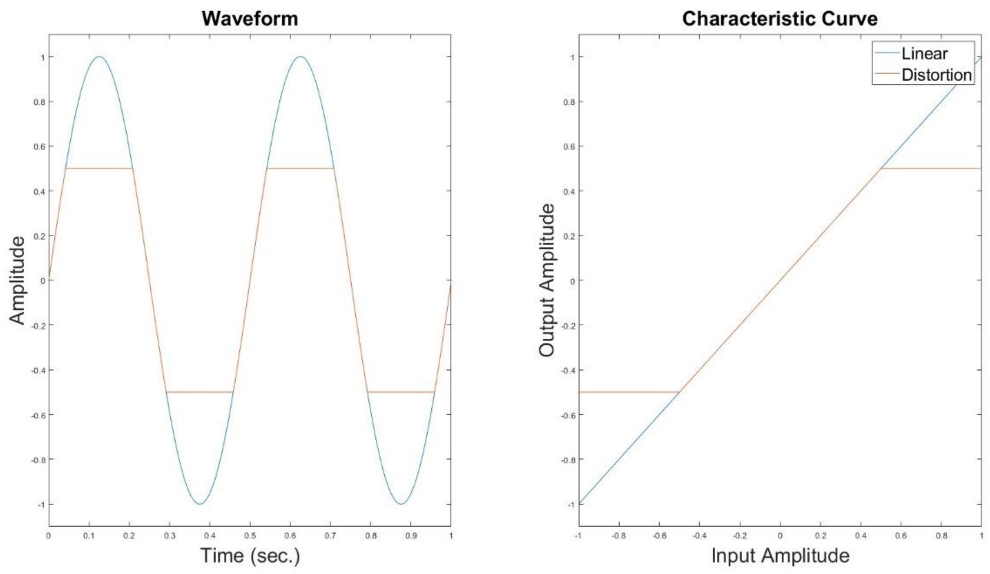


Figure 1.5 The Waveform and Characteristic Curve of Hard Clip Distortion

Distortion Equalisation (Tone Stage)

The next segment of the Big Muff Pi pedal is the inclusion of a potentiometer to control the tone of the outputted signal. From figure 1.2 above, it is shown that a low pass and high pass filter are used in cascading series to change the tone. A potentiometer is used to mix the filters from ‘Full Bass’ to ‘Full Treble’ (ElectroSmash, n.d.). Figure 1.6 shows the frequency response of the tone stage with the potentiometer moved between values.

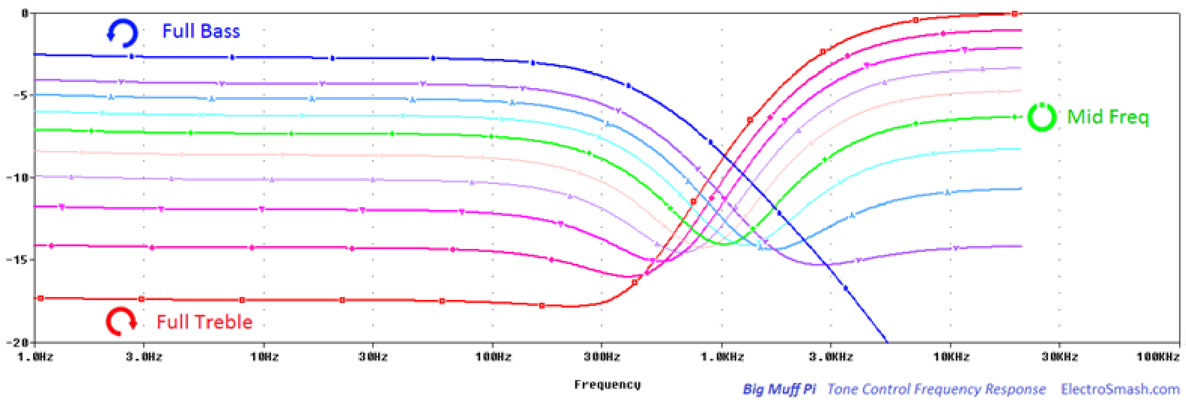


Figure 1.6 The Frequency Response of Different Potentiometer Tone Values (ElectroSmash, n.d.)

It is noted that there is not flat response in this tone stage, the filters create a scoop at 1Khz. Future versions of the Big Muff Pi include a bypass toggle so that a flat frequency response can be observed (Rae, 2012).

Infinite Impulse Response Filters

An infinite impulse response filter utilises the past output and input from a system, as well as the current input (Tan, 2008, pp.304). This allows for less computation power in comparison to finite impulse response filters.

The output of a digital filter system can be shown in equation 4 below.

$$y(k) = x(k)H(k)$$

Equation 4. The Relationship of a Digital Filter and Digital Signal

Where $H(k)$ is the digital filter system response, $x(k)$ is the Discrete Fourier Transform of the input signal $y(k)$. The Discrete Fourier Transform of a signal $x(n)$ can be found using equation 5 below.

$$X(K) = \sum_{n=0}^{N-1} x\,(n)e^{\frac{-j2\pi kn}{N}}, \qquad k = 0,1,\cdots,N-1$$

Equation 5. The Discrete Fourier Transform of a Signal (Tan, 2008, pp.92)

The Discrete Fourier Transform assumes the frequency values to be discrete so that finite values can be computed. By moving to the frequency domain, certain frequencies can be manipulated with filters.

For representation of filter signal flow diagrams, the Z transform is used commonly. This is to highlight the characteristics of a filter, as well as provide suitable block diagrams and difference equations. Equation 6 is the formula for Z transform.

$$x(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

Equation 6. The Z Transform Equation (Tan, 2008, pp.135)

Where $x(z)$ is the Z transform of the input $x(n)$ and z^{-n} is a complex variable. From this, we can equate a system response of an IIR filter represented in the Z-domain, shown in equation 7. This can further be written in the discrete domain with a difference equation shown in equation 8.

$$\frac{y(z)}{x(z)} = H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \cdots + b_Mz^{-M}}{1 + a_1z^{-1} + a_2z^{-2} + \cdots + b_Nz^{-N}}$$

Equation 7. A Digital IIR Filter Transfer Function (Tan, 2008, pp.303)

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \cdots + b_Mx[n-M] + a_1y[n-1]$$
$$+a_2y[n-2] + \cdots + a_Ny[n-N]$$

Equation 8. The Difference Equation of a Digital IIR Filter (Tan, 2008, pp.303)

Where b and a are the numerator and denominator coefficients and M & N are the number of zeros (number of values for which $H(z)$ is infinite) and poles (number of values for which $H(z)$ is infinite) respectively.

Implementation of IIR Filters

The Big Muff Pi utilises two capacitors for the high and low pass filters. This first order design creates a slow roll off characteristic.

A second order IIR filter is used for this implementation, a block diagram for this is shown in figure 1.7. This differs from the physical pedal but allows for the frequency response curve to have a narrower transition band.

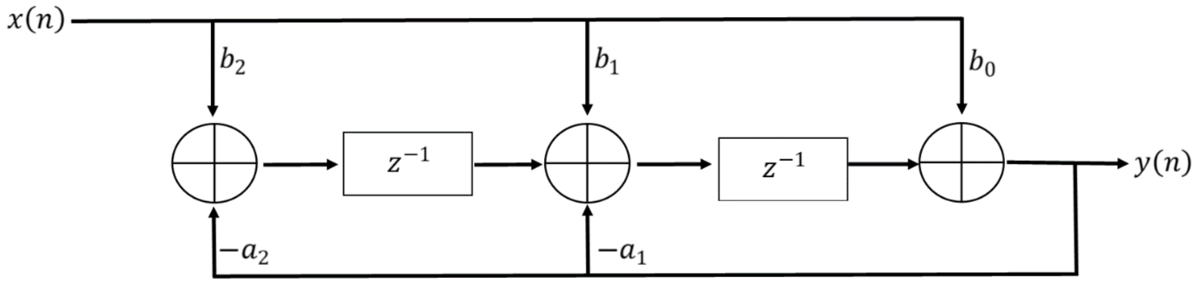


Figure 1.7 A Second Order IIR Filter Block Diagram

An analog prototype can be used to create a high pass and low pass filter.

Analog Modelling

An approach to designing a digital filter is to create the analog counterpart and transform using a series of steps (figure 1.8). Bilinear transformation of the analog filter creates the discrete time filter system from the analog design. This preserves the stability of a system and maps the points from the continuous time filter frequency response. Figure 1.9 below defines the conversion from analog filter specifications to the lowpass prototype.

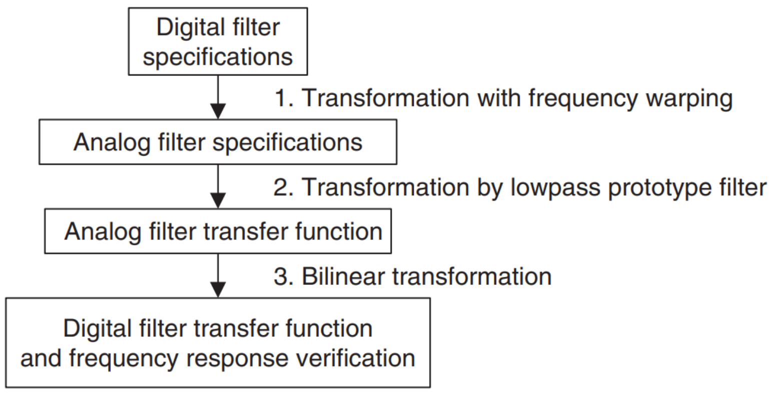
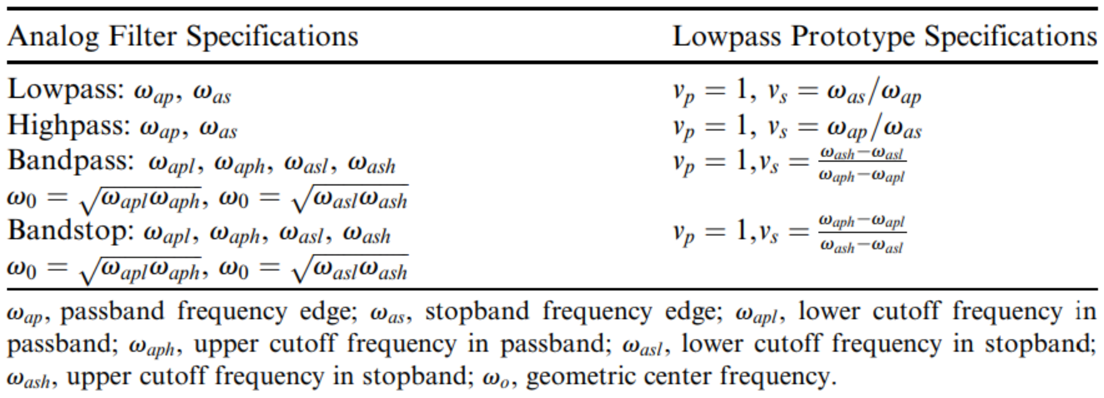

Digital filter specifications
1. Transformation with frequency warping
Analog filter specifications
2. Transformation by lowpass prototype filter
Analog filter transfer function
3. Bilinear transformation
Digital filter transfer function and frequency response verification

Figure 1.8 The Design Method of a Filter (Tan, 2008, pp.305)



Analog Filter Specifications	Lowpass Prototype Specifications
Lowpass: ω_{ap}, ω_{as}	$v_p = 1,\ v_s = \omega_{as}/\omega_{ap}$
Highpass: ω_{ap}, ω_{as}	$v_p = 1,\ v_s = \omega_{ap}/\omega_{as}$
Bandpass: ω_{apl}, ω_{aph}, ω_{asl}, ω_{ash} $\omega_0 = \sqrt{\omega_{apl}\omega_{aph}}$, $\omega_0 = \sqrt{\omega_{asl}\omega_{ash}}$	$v_p = 1, v_s = \frac{\omega_{ash}-\omega_{asl}}{\omega_{aph}-\omega_{apl}}$
Bandstop: ω_{apl}, ω_{aph}, ω_{asl}, ω_{ash} $\omega_0 = \sqrt{\omega_{apl}\omega_{aph}}$, $\omega_0 = \sqrt{\omega_{asl}\omega_{ash}}$	$v_p = 1, v_s = \frac{\omega_{aph}-\omega_{apl}}{\omega_{ash}-\omega_{asl}}$

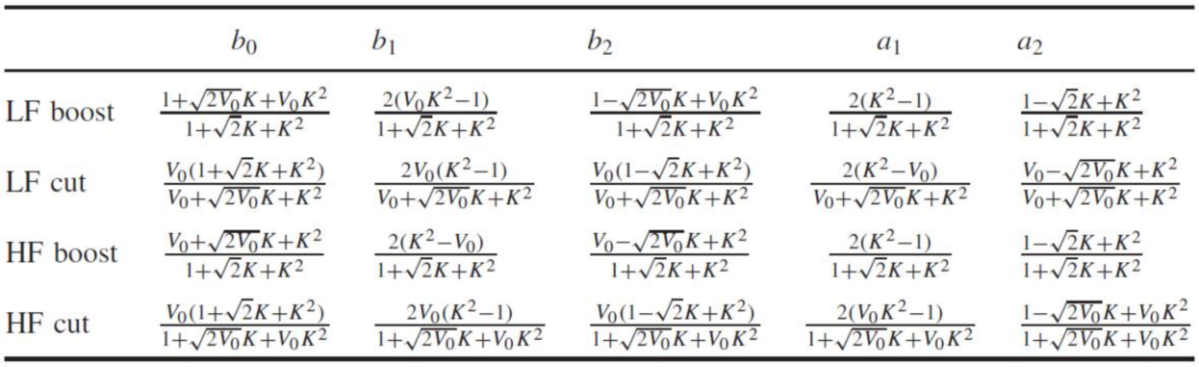
ω_{ap}, passband frequency edge; ω_{as}, stopband frequency edge; ω_{apl}, lower cutoff frequency in passband; ω_{aph}, upper cutoff frequency in passband; ω_{asl}, lower cutoff frequency in stopband; ω_{ash}, upper cutoff frequency in stopband; ω_o, geometric center frequency.

Figure 1.9 The Conversion from Analog to Lowpass Prototype (Tan, 2008, pp.325)

The advantages of this design is not moving into the frequency domain via the Discrete Fourier Transform (Equation 5). This is less cost on a system as Bilinear transformation allows for the movement into the Z domain.

Shelving Filters

For the tone control, shelving filters are used to boost or cut a frequency range specified by the parameters of the IIR filter specifications. Shelving filters boost or cute the low or high frequency bands with the parameters the cut off frequency f_c and gain G (Zolzer, 2011, pp.61). For the Big Muff digital recreation, the cut off frequency for both low and high pass filters are set to 1000hz. This allows for similar frequency response to the physical pedal. Figure 1.10 highlights filter coefficients for 2nd order shelving filter.



	b_0	b_1	b_2	a_1	a_2
LF boost	$\frac{1+\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
LF cut	$\frac{V_0(1+\sqrt{2}K+K^2)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2V_0(K^2-1)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{V_0(1-\sqrt{2}K+K^2)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{2(K^2-V_0)}{V_0+\sqrt{2V_0}K+K^2}$	$\frac{V_0-\sqrt{2V_0}K+K^2}{V_0+\sqrt{2V_0}K+K^2}$
HF boost	$\frac{V_0+\sqrt{2V_0}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-V_0)}{1+\sqrt{2}K+K^2}$	$\frac{V_0-\sqrt{2V_0}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2(K^2-1)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
HF cut	$\frac{V_0(1+\sqrt{2}K+K^2)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2V_0(K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{V_0(1-\sqrt{2}K+K^2)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{2(V_0K^2-1)}{1+\sqrt{2V_0}K+V_0K^2}$	$\frac{1-\sqrt{2V_0}K+V_0K^2}{1+\sqrt{2V_0}K+V_0K^2}$

Figure 1.10 Filter Coefficients for 2nd Order Shelving Filters (Zolzer, 2011, pp.64)

Where K is the frequency parameter for boosting the filters (equation 9) and v_0 is the gain parameter for the filter (equation 10).

$$K = tan\left(\frac{\pi f_c}{f_s}\right)$$

Equation 9. Frequency Parameter for Cutting or Boosting of the Filters

$$v_0 = 10^{G/20}$$

Equation 10. The Gain Parameter for the Filters

Gain Control (Output Stage)

Due to the nature of tone control, volume loss is observed from the original signal. In the pedal, the compensation gain is 13dB (ElectroSmash, n.d.). Output impedance is also dependent on the users placement of the volume potentiometer.

For the digital model, a gain control is added to the design for ease of volume control. Equation 11 is the gain control, g used as a multiplier on the inputted signal from the tone control stage.

$$y(n) = g * x(n)$$

Equation 11. The Volume Control Equation

Each signal value is multiplied by the gain set by the user, which has a range from 0 to 0.99 to avoid a feedback loop.

2. Methodology

The methodology focuses on the implementation of a Big Muff Pi guitar pedal digital model in MATLAB as MATLAB scripts and functions, as well as discussing the completed MATLAB Application. The implemented MATLAB script (main) works as a prototype file for the MATLAB application (BigMuffApp), working with the same parameters for each portion, but with further control and an outputted frequency response for files. Figure 2.1 is a flowchart for the MATLAB script.

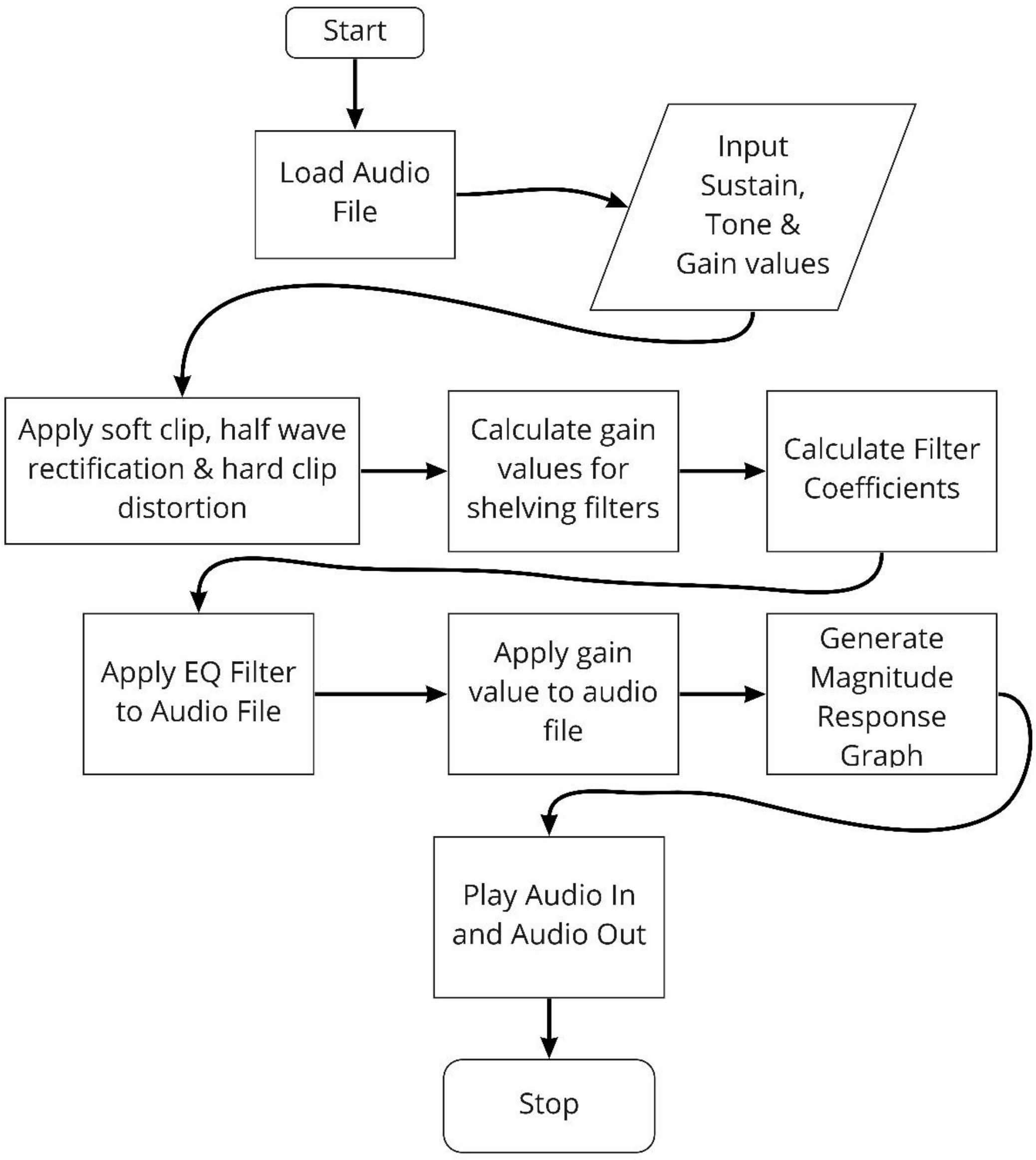
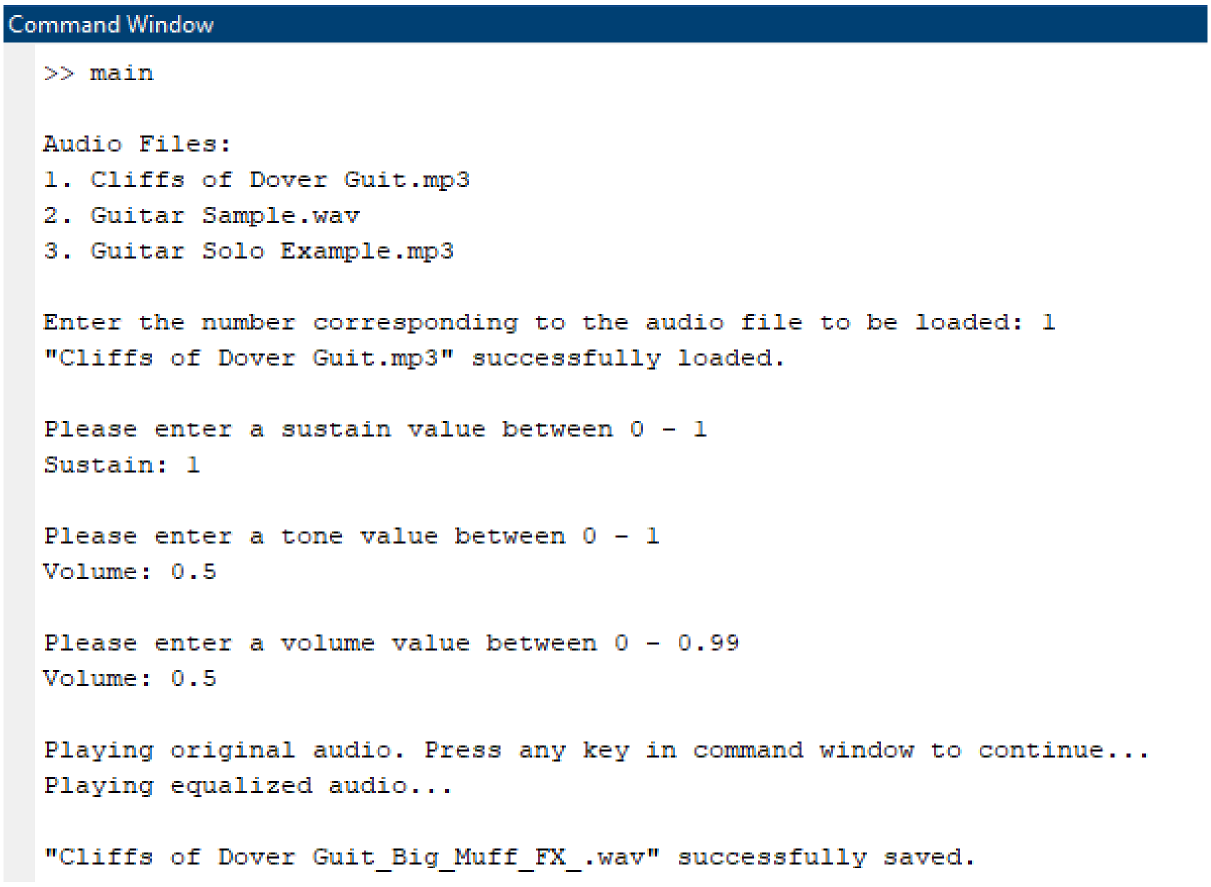

Start
Load Audio File
Input Sustain, Tone & Gain values
Apply soft clip, half wave rectification & hard clip distortion
Calculate gain values for shelving filters
Calculate Filter Coefficients
Apply EQ Filter to Audio File
Apply gain value to audio file
Generate Magnitude Response Graph
Play Audio In and Audio Out
Stop

Figure 2.1 A Flowchart Overview of the Big Muff Digital Model

MATLAB Audio Process

For the MATLAB script, the user is given control of the same controls that are available on the original analog pedal. They are prompted to assign values between parameters, which checks to avoid feedback and overloading issues. This is translated into knob controls for the application. Figure 2.2 is the command line prompts for the script and figure 2.3 below is the corresponding graphic user interface for the application. Three guitar audio files are given with this implementation as examples.



```
>> main

Audio Files:
1. Cliffs of Dover Guit.mp3
2. Guitar Sample.wav
3. Guitar Solo Example.mp3

Enter the number corresponding to the audio file to be loaded: 1
"Cliffs of Dover Guit.mp3" successfully loaded.

Please enter a sustain value between 0 - 1
Sustain: 1

Please enter a tone value between 0 - 1
Volume: 0.5

Please enter a volume value between 0 - 0.99
Volume: 0.5

Playing original audio. Press any key in command window to continue...
Playing equalized audio...

"Cliffs of Dover Guit_Big_Muff_FX_.wav" successfully saved.
```

Figure 2.2 The User Inputs for MATLAB Script Big Muff Digital Emulation

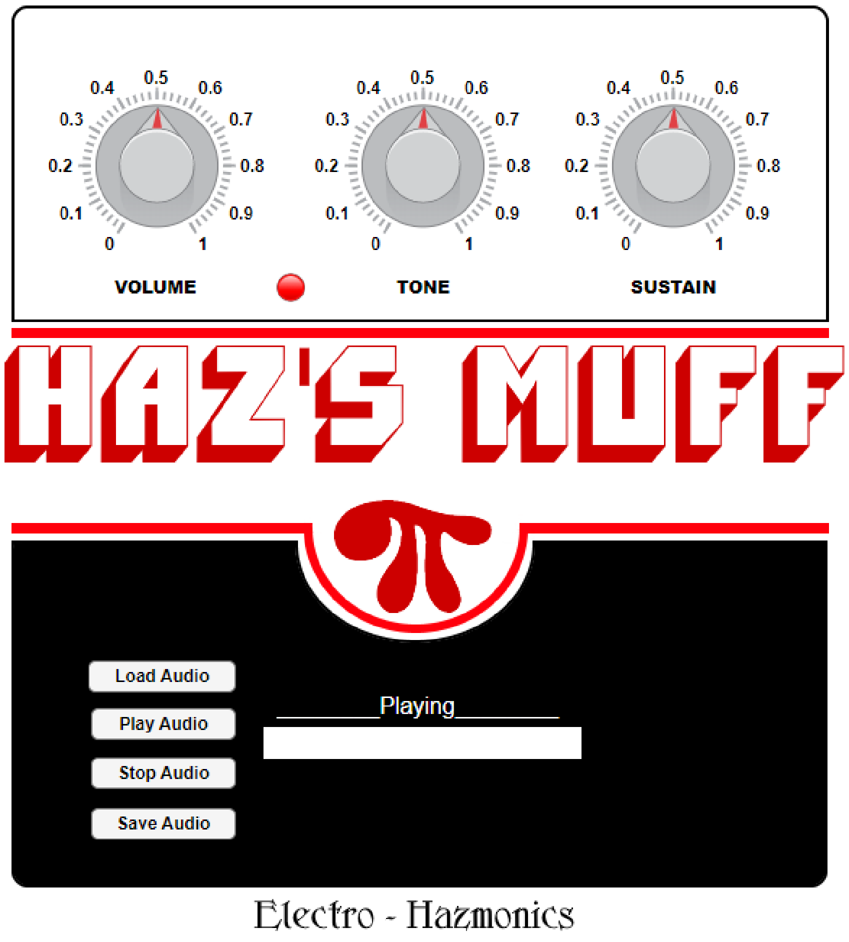


Figure 2.3 The Big Muff Application Graphic User Interface

The application graphic user interface is created to closely resemble the original physical pedal. As the application uses an audio buffer in non real time, the original on/off switch is replaced with various options to load, play and save audio. If recreated in VST3 format, the original on/off switch can be reinstated. The red LED control changes colour to green upon the playing of an audio file.

Distortion Implementation

For the clipping stages, the distortions are applied in series. The soft clip is the first distortion, followed by half wave rectification and a hard threshold clip at the end. By placing the distortion in series, the outputted signal provides a similar distortion effect that occurs in the circuitry of the analog pedal. Future revisions may look at changing the signal to be affected via parallel processing for further changes to the final distortion signal. Figure 2.4 is a block diagram of the clipping stages.

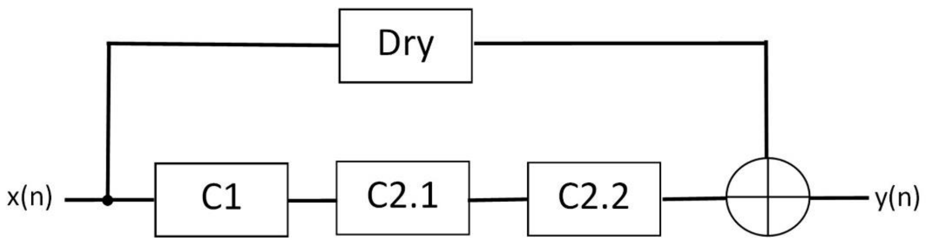


Figure 2.4 A Block Diagram Breakdown of the Distortion Process

Where each block is a distortion effect. C1 signifies the soft clipping cubic distortion and C2.1 is half wave rectification. C2.2 is the hard clipping. From this diagram, we see that the signal is mixed with the dry signal. The signal is duplicated and the sustain control is used to control the mix of both signals.

From this implementation of the clipping, the filtering process can be implemented with the high and low pass 2nd order shelving filters.

Filtering

The filter stage is made with a cascade of filters, with the product of the filters used to change the values of the filter parameters. It is implemented with the aforementioned 2nd order shelving filters, using two shelving filters (high and low pass). The filter coefficients were calculated using figure 1.10 above.

The filters are designed to calculate each individual magnitude response separately and then multiply them together for the product of the equaliser's magnitude response. The inbuilt 'freqz' function is used in this instance using the filter coefficients as inputs, calculating the overall filter response by multiplying all filter responses. Using the inbuilt 'invfreqz' function finds the filter coefficients of the shelving filters. These coefficients can then be placed into the filter() function alongside the input signal to create the affected output signal.

Convolving the individual magnitude responses would lead to the same output, with each convolution leading to the coefficients for the filter() function directly.

Volume

The gain control is added to this project to give the user control over the final volume of the outputted track. Figure 2.5 is a block diagram for this process. x(n) is the input signal from the previous filtering process. y(n) is the final outputted signal for this guitar pedal emulation.

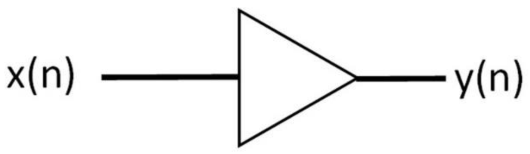


Figure 2.5 A Block Diagram for the Volume Control Section

This gain control affects the whole signal after the filtering process. For further gain control, and extra wet/dry mix gain could be added before the summation of both signals during the distortion process.

3. Results & Discussion

The big muff emulation was tested with outcomes for various minimum, maximum and real world outcomes. comparisons were made during the testing phase with a physical version of the big muff pedal, with the same audio tracks for fair comparison. Figures ?.? below highlights the frequency response of the Tone control at various levels.

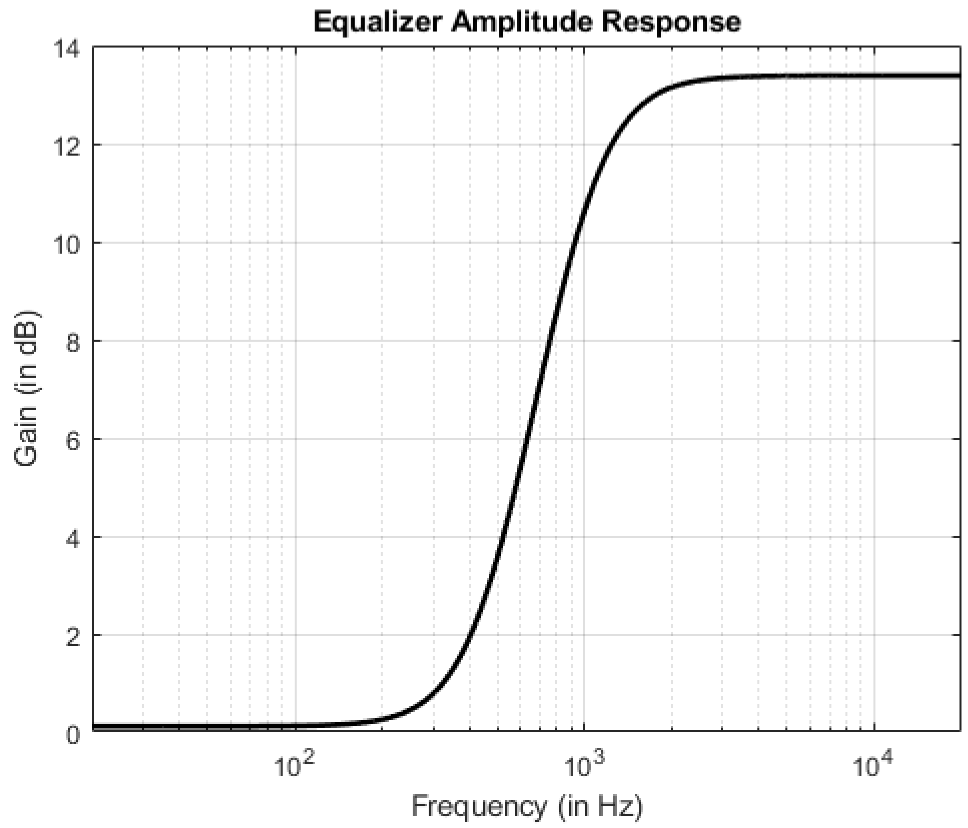


Figure 3.1 The Frequency Response With Tone at 1

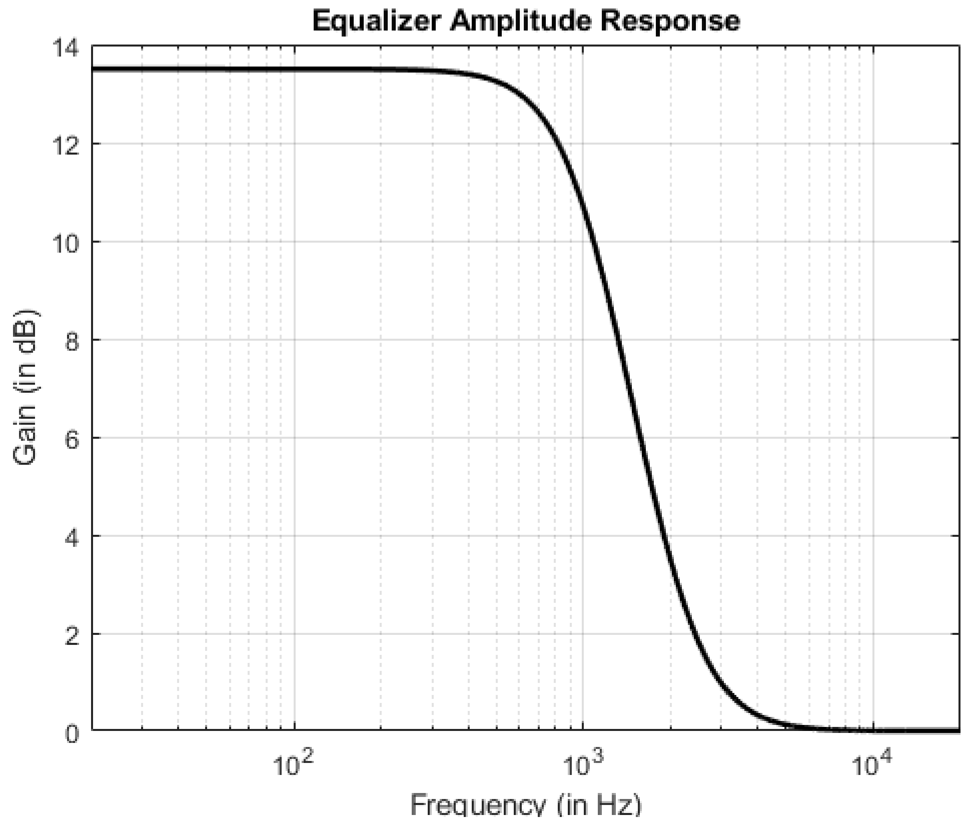


Figure 3.2 The Frequency Response with Tone at 0

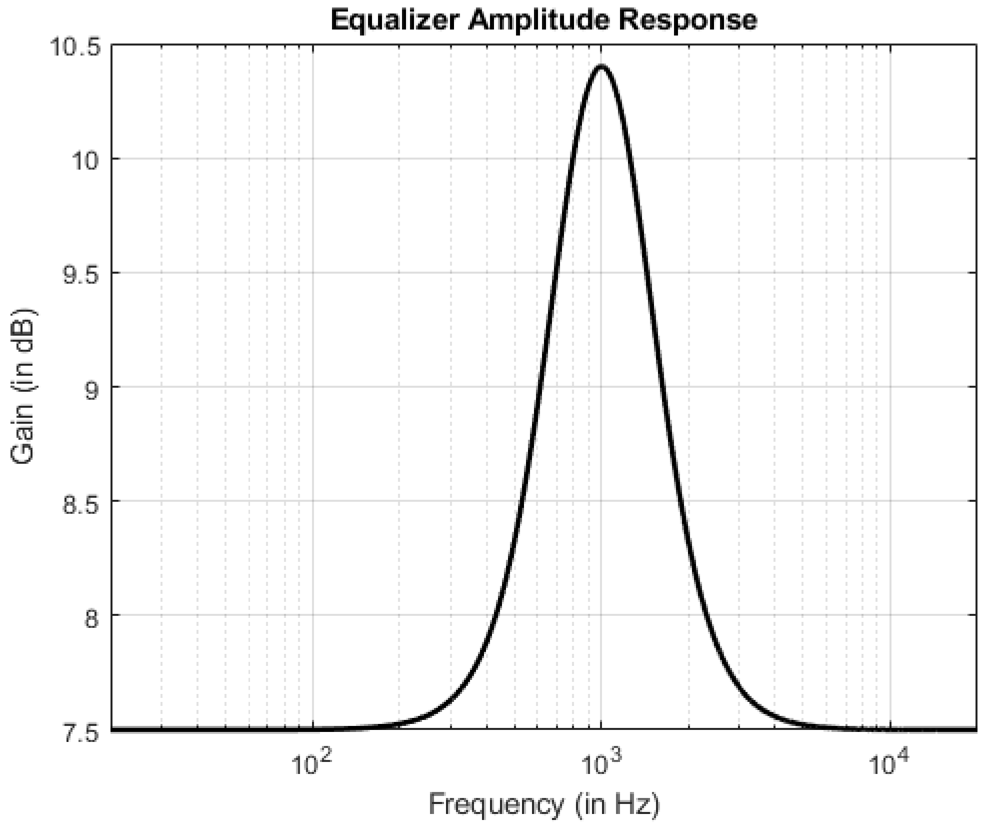


Figure 3.3 The Frequency Response With Tone at 0.5 (midpoint)

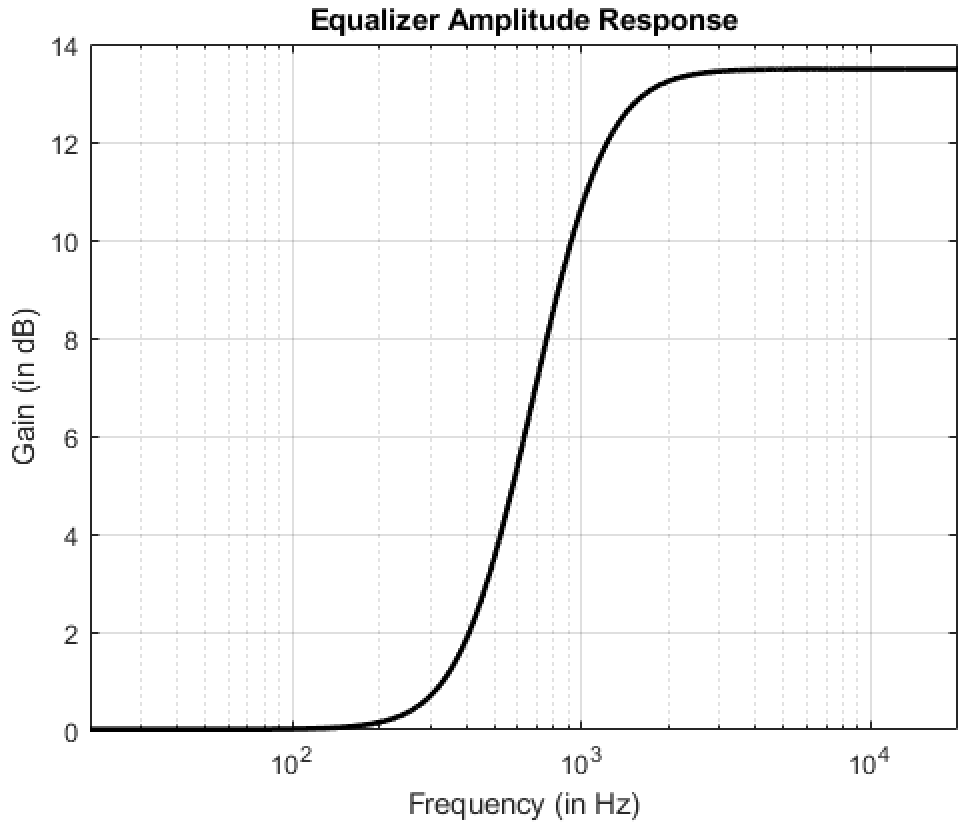


Figure 3.4 The Frequency Response with Tone at 0.7

Discussion and Critical Analysis

The above figures highlight the Tone control implementation within the Big Muff project. By varying the value of the tone, the gain of the peaking filters are changed. This is done by normalising the values depending on the number of potentiometer values. Auditory results from the tone change suggests a functional implementation. In comparison to the physical, moving the value below 0.5 provides a similar, bass heavy tone. Increasing beyond 0.5 gave the treble tone associated with the tone control. When compared to the frequency response from ElectroSmah (n.d), the implementation slightly differs. This is due to the nature of the circuitry. The tone filter is used to boost values that have been lost due to the resistance in the circuit.

Critically analysing this, the implementation, while successful, lacks the originality of the physical pedal. The order of the IIR filter is also changed, which affects the outputted signal. Future versions of the tone control should implement a filter that closely resembles the original.

The designed distortion phase is a combination of the original circuitry and the new addition of the half wave rectification. This was to create the overdriven sound and relay the original nature of the pedal. Tests with solely soft and hard clipping were undertaken. Upon critical reflection, the distortion phase could take a closer resemblance to the pedal by using solely the soft and hard clip, but with parallel processing. However, the combination created provides a satisfactory distortion and reduces computing time with cascading features.

The volume control solely affects the signal after the wet/dry mix. This is to correspond to the original implementation. However, due to the lack of gain loss in digital audio, this stage may benefit from future revisions, such as a wet/dry mixer and pre distortion gain control.

The auditory result from the MATLAB application provides a satisfactory similarity to the Electro Harmonix Big Muff Pi pedal. While choices within filtering and distorting differ, it is noted that a satisfactory tone that takes inspiration from the pedal is observed. It is however, not a real time application.

A future implementation of this digital pedal would be to recreate within the C++ programming language with JUCE framework. This would allow for a real time audio buffer and the creation of a VST3 plugin, which can be used within a guitarist’s digital audio workspace.

4. Reference List

Electro Harmonix (2022). *Big Muff Pi | Fuzz / Distortion / Sustainer - Electro-Harmonix*. [online] https://www.ehx.com/. Available at: https://www.ehx.com/products/big-muff-pi/ [Accessed 17 Jun. 2022].

ElectroSmash (n.d.). *ElectroSmash - Big Muff Pi Analysis*. [online] www.electrosmash.com. Available at: https://www.electrosmash.com/big-muff-pi-analysis [Accessed 18 Jun. 2022].

Holters, M., Dempwolf, K. and Zolzer, U. (2011). A Digital Emulation of the Boss SD-1 Super Overdrive Pedal Based on Physical Modeling. In: *Audio Engineering Society Convention Paper 8506*. AES 131st Convention. New York.

Malaker, E. (2019). *Soft Clipping Vs Hard Clipping – What Is the Difference?* [online] Humbucker Soup. Available at: https://humbuckersoup.com/soft-clipping-vs-hard-clipping-difference/ [Accessed 17 May 2022].

Pirkle, W.C. (2019). *Designing Audio Effect Plugins in C++ : for AAX, AU, and VST3 with DSP Theory*. New York: Routledge, pp.15–30, 535–563.

Rae, K. (2012). *Big Muff Transistors and Other Components*. [online] The Big Muff Pi Page. Available at: http://www.kitrae.net/music/big_muff_guts.html#Circuit [Accessed 17 Jun. 2022].

Reiss, J.D. and McPherson, A.P. (2015). *Audio Effects theory, Implementation and Application*. London: Taylor & Francis Group, pp.167–188.

Russell, A. (2020). *The Complete Guide to Distortion and Saturation*. [online] EDMProd. Available at: https://www.edmprod.com/distortion-saturation-guide/ [Accessed 24 May 2022].

Tan, L. (2008). *Digital Signal Processing : Fundamentals and Applications*. Amsterdam: Academic Press, pp.87–98, 303–412.

Tarr, E. (2019). *Hack Audio: an Introduction to Computer Programming and Digital Signal Processing in MATLAB*. London: Routledge.

Zolzer, U. (2011). *DAFX : Digital Audio Effects*. Chichester, West Sussex, England: Wiley.

Appendix 1. MATLAB Scripts & Functions for Distortion Examples

```
% DISTORTIONEXAMPLE
% This script is used to test various distortion functions.
% Each algorithm can be analyzed by "uncommenting" the
% code under each section. The waveform, characteristic curve,
% and total harmonic distortion (THD) is plotted for each function.

clear;clc;

Fs = 48000;
Ts = 1/Fs;
f = 2; % f = 2 (Waveform and Char Curve), f = 2500 (THD)
t = [0:Ts:1].';
in = sin(2*pi*f*t); % Used as input signal for each distortion


%%% Half-wave rectification
out = halfwaveRectification(in);

%%% Full-wave rectification
%out = fullwaveRectification(in);

%%% Hard-clipping
%thresh = 0.5;
%out = hardClip(in,thresh);

%%% Cubic soft-clipping
%a = 1; %Amount: 0 (no distortion) - 1 (full)
%out = cubicDistortion(in,a);

%%% Plotting
figure(1); % Use f = 2 (above)
subplot(1,2,1); % Waveform
plot(t,in,t,out); axis([0 1 -1.1 1.1]);
xlabel('Time (sec.)','FontSize',24);ylabel('Amplitude','FontSize',24);
title('Waveform','FontSize',24);

subplot(1,2,2); % Characteristic curve
plot(in,in,in,out); axis([-1 1 -1.1 1.1]);
xlabel('Input Amplitude','FontSize',24); ylabel('Output Amplitude','FontSize',24);
legend('Linear','Distortion', 'FontSize',20); title('Characteristic
Curve','FontSize',24);

figure(2); % Total harmonic distortion plot (f = 2500)
thd(out,Fs,5);
axis([0 24 -50 0]);
```

```matlab
% Cubic Distortion
% This function implements soft clipping
% distortion. An input parameter "a" is used
% to control the amount of distortion applied
% to the input signal

% Input variables
%   in : input signal
%   a : drive amount (0-1). amplitude of 3rd harmonic

function [ out ] = cubicDistortion(in,a)
N = length(in);
out = zeros(N,1);
for n = 1:N
out(n,1) = in(n,1) - a*(1/3)*in(n,1)^3;
end
```

```matlab
% HALFWAVERECTIFICATION
% This function implements full-wave rectification
% distortion. Amplitude values of the input signal
% which are negative are changed to zero in the
% output signal.

function [out] = halfwaveRectification(in)

N = length(in);
out = zeros(N,1);
for n = 1:N

    if in(n,1) >= 0
        % If positive, assign input to output
        out(n,1) = in(n,1);
    else
        % If negative, set output to zero
        out(n,1) = 0;

    end

end
```

```matlab
% Hard Clip
% This function implements hard-clipping
% distortion. Amplitude values of the input signal
% that are greater than a threshold are clipped.

% Input variables
%   in : signal to be processed
%   thresh : maximum amplitude where clipping occurs

function [out] = hardClip(in,thresh)
%HARDCLIP Summary of this function goes here
%   Detailed explanation goes here
N = length(in);
out = zeros(N,1);
for n = 1:N

    if in(n,1) >= thresh
        % if true, assign output = thresh
        out(n,1) = thresh;
    elseif in(n,1)<= -thresh
        % if true, set output = -thresh
        out(n,1) = -thresh;
    else
        out(n,1) = in(n,1);
    end

end
```